```
/* Method of Steepest Descent. This code is written for Example 9.2 of the text book.*/
/* However, it works for any problem by changing the function */

#include <stdio.h>
#include <math.h>
#define  MACHEPS  1e-15
double  D[3], Y[3];
double  X[11],X1[11];
double  e,xk;
int i,l,m,n;



/*  Function subroutine with derivatives    */

/*  Prob. 9.1 */
/*  double Eval()
{
  D[1]=(2.0*X[1])+X[2]; D[2]=(4.0*(X[2]))+X[1];
  return X[1]*X[1]+2.0*X[2]*X[2]+X[1]*X[2];
}*/

double Eval()
{
  D[1]=(4.0*X[1]*X[1]*X[1])+(4*X[1]*X[2])-(42*X[1])+(2*X[2]*X[2])-14;
  D[2]=(4.0*X[2]*X[2]*X[2])+(2*X[1]*X[1])-(26*X[2])+(4*X[1]*X[2])-22;
  return (X[1]*X[1]*X[1]*X[1])+(2*X[1]*X[1]*X[2])-(22*X[2])-(22*X[1]*X[1])+(X[2]*X[2])+121+(X[2]*
X[2]*X[2]*X[2])+49+(2*X[1]*X[2]*X[2])-(14*X[1])-(14*X[2]*X[2])+(X[1]*X[1]);
}

  // Function called by Steepds()
  void Utilit()
  {
    int i;
    double dd;
    // Find the magnitude of the gradient
    dd=0.0;
    //for (i=1; i<l+1; i++)
    dd = D[1]/D[2];
    if(D[1]<0)
    {
        X1[1]=X[1];
      X[1] = X[1]+(xk*dd);
    }
    else
        {
        X1[1]=X[1];
      X[1] = X[1]-(xk*dd);
    }


    if(D[2]<0)
    {
        X1[2]=X[2];
      X[2] = X[2]+(xk);
    }
    else
    {
        X1[2]=X[2];
      X[2] = X[2]-(xk);
    }
    Y[2]=Eval();
  }


/*INPUTS:
   l - The dimension of function to study
   e - The convergence criteria
   m - The maximum number of iterations
   xk - A starting constant
   X[i] - Initial values of variables

OUTPUTS:
   X[i] - The locally optimum set
   Eval - The value of local maximum
   n - The number of iterations performed,*/
```

```
  void Steepds()
  {
   int i;
  n=0;
     Y[1]=Eval();
     Utilit();
 e50: if (fabs(Y[2]-Y[1])<MACHEPS) goto e51;
  if ((Y[1]-Y[2])>0.0) xk=xk;

e51: if (Y[2]>Y[1]) xk=xk;

  if (Y[2]<Y[1]) goto e100;
  // Restore the X[i]
  for (i=1; i<l+1; i++)
  {
    X[i]=X1[i];
  }
  xk=xk/2.0;
     goto e200;
e100: Y[1]=Y[2];
  // Obtain new values
e200: Y[1]=Eval();
  // Update X[i]
  Utilit();
  // Check for maximum iterations and convergence
  n++;
  if (n>=m) return;
    if (fabs(Y[2]-Y[1])<e) return;

  // Try another iteration
  goto e50;
} // Steepds()


void main() {
  printf("\n How many dimensions: "); scanf("%d",&l);
  printf("\n Accuracy: "); scanf("%lf",&e);
  printf("\n Maximum number of iterations: "); scanf("%d",&m);
  printf("\n Starting step size for second variable (X2): "); scanf("%lf",&xk);
  printf("\n Initial guess for both variables:\n\n");
  for (i=1; i<l+1; i++) {
    printf("     X(%d) = ",i); scanf("%lf",&X[i]);
  }

  Steepds();   // Call steepest descent optimization

  printf("\n\n The results are:\n\n");
  for (i=1; i<l+1; i++) {
    printf("     X(%d) = %1.7f\n",i,X[i]);
  }
  printf("\n Minimum found = %2.7f\n",Eval());
  printf("\n The number of iterations was %d\n\n",n);
}

// End of file
```